CLAIMS

1. A process for automatically producing software for a computer using a plurality of components which exist in executable code, comprising the steps of:

providing an input interface for each of the components in which respective methods of each of the components are defined which can be called and implemented as part of the respective component;

providing an output interface for each of the components in which respective data formats are defined for data of a respective event as a result of implementation of one of a respective method and a respective component, and in which respective further methods are defined which can be called in the respective component but are not executably contained in the respective component;

depicting, in a graphical editor, a symbol corresponding to one of the components having a respective input interface and a respective output interface;

offering a selection option for directional linking of an output interface of one of the components to an input interface of another of the components; and

producing a program code linking the one component to the another component based on links made.

2. A process for automatically producing software for a computer as claimed in Claim 1, further comprising at least one of the following steps:

calling a respective method, via the program code and based on a respective event, which is defined in the input interface of the another component, and transferring, via the program code, the data of the respective event to the respective method which is called, the data being expected by the method which is to be called; and

converting, via the program code, the respective data formats of the callable methods in the output interface of the one component into the respective data formats of the available methods of the input interface of the another component, and converting the methods into one another.

3. A process for automatically producing software for a computer as claimed in Claim 1, further comprising at least one of the following steps:

comparing the definition of the method to be called in the output interface of the one component with the definition of available methods of the input interface of the another component; and

comparing the respective data formats of an event of the output interface of the one component with the respective data formats to be transferred to a method of the input interface of the another component.

4. A process for automatically producing software for a computer as claimed in Claim 3, further comprising at least one of the following steps:

matching the data formats of the callable methods in the output interface of the one component to the data formats of the available methods of the input interface of the another component; and

matching the data formats of the event of the output interface of the component to the data formats to be transferred to the method of the input interface of the another component if they are not compatible.

5. A process for automatically producing software for a computer as claimed in Claim 1, wherein a link from one of an event and a method of the output interface of the one component to a plurality of methods of the input interface of the another component can be chosen.

6. A process for automatically producing software for a computer as claimed in Claim 5, further comprising the step of:

determining a condition for selecting the methods of the input interface of the another component for the link.

7. A process for automatically producing software for a computer as claimed in Claim 1, wherein an input interface and an output interface belong to the same component.

8. A process for automatically producing software for a computer as claimed in Claim 1, wherein a plurality of links can be made for a method of an input interface.

9. A process for automatically producing software for a computer as claimed in Claim 1, wherein a component can be represented a plurality of times as a symbol.

10. A process for automatically producing software for a computer as claimed in Claim 1, wherein the symbols for components can be arranged freely on a display area of the graphical editor.

11. A process for automatically producing software for a computer as claimed in Claim 1, further comprising the steps of:

producing and compiling the program code in a compiler language; and

associating the produced and compiled program code with the components to form an executable program.

12. A process for automatically producing software for a computer as claimed in Claim 1, wherein the program code is produced in an interpretator language.

13. A process for automatically producing software for a computer as claimed in Claim 12, wherein the interpretator language is XML.

14. A process for automatically producing software for a computer as claimed in Claim 12, further comprising the steps of:

combining the program code with the components as a dynamic link library; and

combining the program code with an interpretator to form an executable complete program.

15. A process for automatically producing software for a computer as claimed in Claim 1, further comprising the step of:

connecting at least two of the components to form a new complete component, it being possible to stipulate which methods and events of the output interfaces of the at least two components used form the output interface of the complete component, and which methods of the input interfaces of the at least two components used form the input interface of the complete component.

16. A process for automatically producing software for a computer as claimed in Claim 1, wherein the software is controlled software for a telecommunications installation.

17. A process for automatically producing software for a computer as claimed in Claim 16, wherein the telecommunications installation is a telephone exchange.

18. A process for automatically producing software for a computer as claimed in Claim 16, wherein the control software is on a control computer in the telecommunications installation.

19. A computer for automatically producing software using a plurality of components which exist in executable code, comprising:

an input interface for each of the components in which respective methods of each of the components are defined which can be called and implemented as part of the respective component;

an output interface for each of the components in which respective data formats are defined for data of a respective event as a result of implementation of one of a respective method and a respective component, and in which respective further methods are defined which can be called in the respective component but are not executably contained in the respective component; and

a graphical editor, wherein a symbol corresponding to one of the components having a respective input interface and a respective output interface is depicted;

wherein a selection option for directional linking of an output interface of one of the components to an input interface of another of the components is offered, and a program code linking the one component to the another component is produced based on links made.